2.19 *pkactn* FORMALISM

Subroutine *pkactn* is the true decision executive at a given decision level. It first enters a value-driven loop which calls subroutines *aslct*, *aproj*, and *aeval* (these are generic subroutine names; actual entry names are passed as arguments to *pkactn*). These subroutines generate, project, and evaluate the alternative. The first pass through the value-driven loop will only consider those alternatives biased in production rules. The second pass considers all allowed alternatives if the first pass generates nothing. If the decision is not value-driven, these routines mimic a value-driven process as follows: *aslct* generates the decision the first time it is called; the second time it is called it sets *more*=.false. and returns. *Aproj* and *aeval* are dummy routines when a mock value-driven decision is made.

aslct Routines

The purpose of the *aslct* routine is to generate an alternative each time it is called, or return *more*=.false. if all appropriate alternatives for consideration have been generated. Because the set of alternatives considered continually changes as the Brawler model is developed, it is not sufficient to label each alternative by a simple numerical index; such a system would greatly reduce the flexibility available for adding new alternatives. Instead, alternatives are labelled with a hierarchical notation consisting of four indexes, the index set (*ilevel, kalt, icall, lcall*). The outermost variable *ilevel* denotes the decision level for which the alternative is a course of action. *Kalt* denotes the most general kind of alternative. For instance, at the maneuver level, *kalt*=4 classifies the alternative as a one-versus-one offensive alternative, while a *kalt* value of 7 indicates a ground-avoidance maneuver. The variable *icall* is used to further differentiate alternatives when several have the same *ilevel* and *kalt* values. Thus, at the maneuver level, when *kalt*=5 (evade a hostile aircraft), *icall*=1 denotes a right break turn and *icall*=5 denotes a maneuver designed to force the threat aircraft to overshoot. The variable *lcall* is used in those cases where a breakdown beyond the *icall* level is required. It is currently used only in the specification of flight tactics.

*aslct*N (N corresponds to *ilevel*) uses the *kalt* index from the index set to break up the alternative enumeration into more easily manageable parts. For each *kalt* value, each call to *aslct*N causes it to call an *alt*NK subroutine (N=*ilevel*, K=*kalt*) until the latter returns *more* = .false., indicating exhaustion of alternatives. The *aslct* routine is responsible for setting *icall* to zero prior to the first time each *alt* routine is called (for each consciousness event). This serves to trigger internal initialization by the *alt* routine. The *alt* routine returns *icall* as the *icall* value of the generated alternative.

aproj Routines

The *aproj* generic subroutine is used to project or predict the consequences of adopting an alternative without putting a value on the consequence. For the maneuver level, for example, the routine *aproj3* predicts the physical relationships between aircraft that would result after implementing a particular trial alternative. For those decisions that are not actually value-driven, the *aproj* subroutine is a dummy.

aeval Routines

The *aeval* generic subroutine places a value on the predicted consequence of a trial alternative. Details of the scoring for each decision level will be discussed below.

akshn Routines

The *akshn* generic subroutine is used to actually implement the action that *pkactn* has selected. This implementation includes altering the value parameters of a conscious pilot, sending orders, and planting weapon launch and maneuver change events.

2.19.1 Functional Element Design Requirements

Design requirements are provided for each decision type in the CMS sections that follow this one. Requirements for alternative projection, evaluation, and selection differ slightly for each type of decision to be simulated.

2.19.2 Functional Element Design Approach

Design approaches are provided for specific decisions in the CMS sections that follow this one. There is not a unique design approach for this function. Rather the alternative selection, projection, and actions simulated are described in the design for each decision making FE.